For a better understanding of Arabic Calligraphy

Nouamane Tazi
CentraleSupélec
nouamane.tazi@student-cs.fr

Asmae Khald
CentraleSupélec
asmae.khald@student-cs.fr

Mustapha Ajeghrir
CentraleSupélec
mustapha.ajeghrir@student-cs.fr

Abstract

In the Arabic heritage and culture, Calligraphy is essential. It is being used for the decoration of houses, public places, and mosques. Each calligraphy style is specific to its author, but there are some shared rules that should be respected. In this work we are going to explore the possible methods in style, character, and words recognition. The first proposed model will infer the calligraphy style, the second will infer the characters from the input image.

1. Introduction

Arabic calligraphy combines the arts of drawing and writing, producing a unique form of textual art. The art originated when Muslims started writing and documenting the Holy Quran [12]. Since then, Arabic calligraphy has been expanded and diversified in styles, producing one of the world's great art forms [5]. This art also holds much historical information, as Arabic calligraphy is used in much of Islamic art and architectural design [6]; therefore, such texts are a very valuable and rich resource for data. There is therefore a practical need to automate the reading of Arabic calligraphy. However, little research has been done in this area, and few resources exist.

1.1. Arabic Language Specification

The machine reading of Arabic presents many interesting challenges . Arabic has 28 main letters, and is written from right to left. The letters are very sensitive to the use of discrete marks to distinguish between them. It is common to have three letters with the same body form, but distinguished with dots or marks placed above or under them (for example,). Arabic letters are also represented by different forms according to their position in the word: start, middle, last, or isolated. This is not the case for the vowel letters $(gj,)$), which have no start or middle location forms because they split words apart as they are not connected to any letters on the LHS (left hand side).

On top of these challenges, the machine reading of Arabic calligraphy has many additional interesting challenges: the variety of styles, the extra level of cursive forms, the interweaving of letters and words, rotations and intersections, etc. Figure 1 show the difference between Arabic text types.

Given all these difficulties for the machine reading of Arabic calligraphy, how is it possible to envisage an AI system that could work? The answer is the restricted text types that use Arabic calligraphy. Arabic calligraphy is not used to write general Arabic texts, but rather to glorify text from the Holy Quran. This restriction greatly simplifies the task and makes it feasible. In Bayesian terms, we know a lot about the prior distribution of calligraphic texts. Through extracting from each text phrase, a bag of letters can be obtained to simplify the indexing of the suggested answer. By comparing the result of letter spotting with the probable bag of letters, the correct text for the selected image can be found.

Figure 1: The difference between Arabic text types in the writing of one word (al-Arabiya).

2. Arabic Calligraphy Style Recognition

2.1. Previous Work

The first research in this area used Linear Discernment Analysis (LDA) as a means of feature extraction from these documents [7]. This research focused on extracting only three Arabic letters (aleph, lam, ain) and compared K-nearest neighbour with Naive Bayes classification obtaining

excellent results. Back-propagation neural network classification was used in [8] to identify the font type of Arabic calligraphy. This study applied image binarisation with edge direction matrices for future extraction, and obtained 43.7% recognition accuracy. The novel approach of Triangle Model feature extraction was used in [6]. Distance-based methods were then used to compare images. Applying Multi-Layer Perceptron and Random Forests as classifiers, respectively gave average accuracy of 50% and 65%.

The previous researches were focused on analysing and classifying the different types of AC styles. From the classification of these styles, a proposed new feature to enable the extraction of text from decoration was explored (Bataineh, Abdullah and Omar, 2011; Azmi and Omar, 2013). Other research has applied the object recognition feature in order to recognise font styles from images (Talab, Abdullah and Razalan, 2013), or has used text descriptors to extract features from AC images (Kaoudja, Kherfi and Khaldi, 2019). This research has used standard machine learning methods in classifying AC styles, with different supervised methods (Bataineh, Abdullah and Omar, 2011; Talab, Abdullah and Razalan, 2013; Kaoudja, Kherfi and Khaldi, 2019) being compared with the unsupervised learning model in Azmi and Omar (2013). Table 2-4 summarises the previously used Arabic calligraphy classifiers in comparison with the machine learning methods that have been used. The table also records for each paper the size of the dataset used, the number of calligraphy styles included, and the average recognition rate recorded for each classifier. The average classification for the styles included is above 90%; this reflects the types of scripts used and the number of training sets for each style.

Paper	*Classifier*	*Size of dataset*	*Number of AC styles*	*Average classification rate*
Bataineh, Abdullah and Omar (2012)	Multi-layer Network Decision Tree	700 images, 100 samples for each style.	7	96.65%
Azmi and Omar (2013)	Multi-layer Perceptron Random Forest compared with Distance Metrics	Total of 1,225 image samples	5	96.67%
Talab, Abdullah and Razalan (2013)	Multi-Layer Neural Network Random Forest	700 images, 100 samples for each style.	7	99%
Kaoudja, Kherfi and Khaldi (2019)	Combination of Multi-layer Perceptron, Support Vector Machine and k-Nearest Neighbour	Total of 1,685 image samples	9	96.44%

Figure 2: Summary of Arabic calligraphy classifiers

2.2. Implementation

Our implementation is highly inspired from [3] and the data (synthetic and real) from [2]. The main problem in ACSR (Arabic Calligraphy Style Recognition) is the scarcity of data. Hopefully, it is possible to generate data using digital fonts to help the training.

2.2.1 Problem Definition

To use already available data, we only focus on 2 style classes : *Ruqaa* and *Farsi*. We plot the a sample of the real images in the figure 3. The goal is the distinguish between the 2 writting styles.[1]

Ruqaa Nastaliq (Farsi)

Figure 3: The 2 classes *Ruqaa* and *Farsi* from real images

2.2.2 Data generation

It is not enough to simply use the basic synthesized data without further processing, the figure 4 shows some differences between basic synthesized images and real images. It is important to note : real images are taken from books.

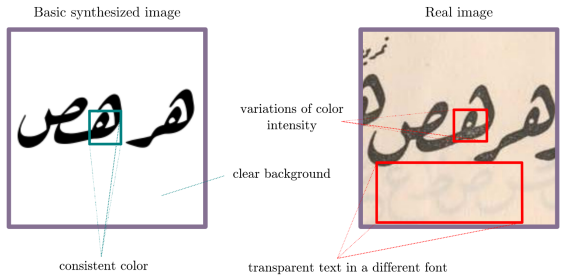


Figure 4: Basic synthesized images against real images

Image generation is done in 4 steps as explained in the figure 5. First, a random background is chosen. Secondly a random word is positioned randomly in the canvas with a random opacity and size. Thirdly, other words a being chosen randomly to represent the paper transparency. Finally, Gaussian and JPEG noise are being introduced.

Using this strategy, $40'000$ images have been generated against 516 real images.

[1] Source code: `https://github.com/Mustapha-AJEGHRIR/arabic_calligraphy`

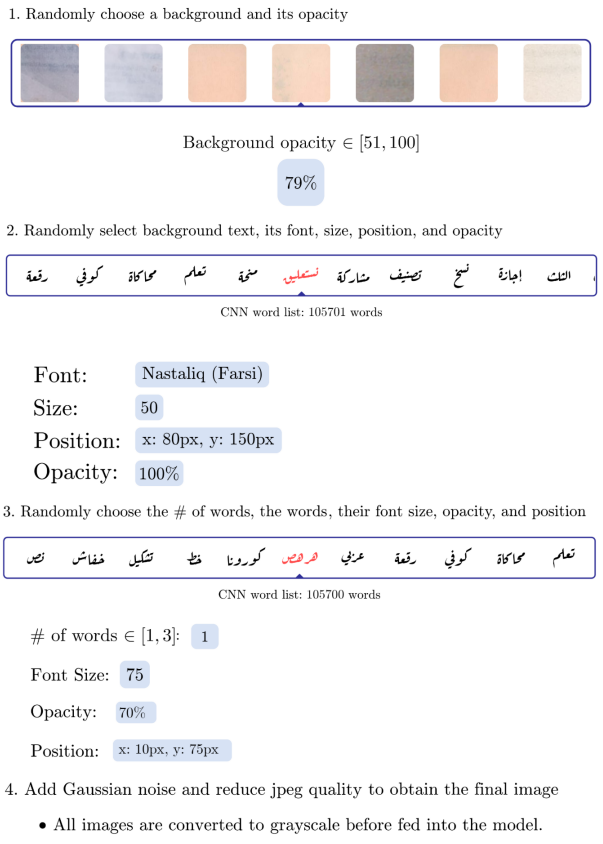


Figure 5: The steps to generate synthetic images

2.2.3 Training strategy and Results

The final used model is in the figure 6. It is mainly composed of Convolutions and Dense layers. The used split is shown in the figure 7. It is important to notice how small the amount of real data is used during the final training (Test mode). This doesn't prevent the model to reach very good results.

Mode	Accuracy
Validation mode	0.99
Test mode	0.97

2.3. Saliency maps

Plotting Saliency maps helps us interpret what the model and understands and how inference was made. The figure 8 plots saliency maps for 6 input images in gray (model input is in gray). We notice that the focus is not made over all the scriptures present in the image, but only over some special curves and letters, this means the model is successfully focusing on the differences between the two classes. In the case where the model focuses on meaningless parts of the image, it should be important to review data biases.

2.4. Demo

A live demo is present in the following HuggingFace spaces : `https://huggingface.co/spaces/mustapha/ACSR`

3. Arabic Calligraphy OCR

3.1. Previous Work

Because of extensive usage of the internet, which provides various media to be analysed, translated or learned, digitisation of different types of information is a significant concern for each culture in order to save itself from being lost and forgotten. A prominent example is the recent Ithaca model by Deepmind [1], the first Deep Neural Network for the textual restoration, geographical and chronological attribution of ancient Greek inscriptions. Reading text from historical images is more challenging than from regular images (Panichkriangkrai et al., 2017). The process of detecting text from source images is well known (text recognition). It has been widely researched and investigated for different languages and various types of text, whether handwritten, machine-printed or in styles of calligraphic art (Bhowmik et al., 2018). The calligraphy domain is more challenging than the other types of text. In comparing the amount of research and number of achievements, calligraphy has seen fewer developments and is not explored as extensively. This results from the challenges in dealing with the different format of the original text as shapes more than letters and words (Nagy, 2017; Panichkriangkrai et al., 2017). Impressive results have been achieved in reading and handling Chinese calligraphic scripts or styles (Jiulong et al., 2017). This is due to the similarity of the letters in this language to calligraphic and drawn scripts. The symbolic style of each character in Chinese is constant; they are not shapes that change according to the position in the word like Arabic letters. Text recognition in different languages (English, French, Chinese and Korean) has been summarised by Ye and Doermann (2015).

In Arabic language recognition, an average accuracy of more than 90% was achieved in correctly digitising handwriting from images (Rabi, Amrouch and Mahani, 2017). Calligraphy is more complicated than handwritten text, and the same approach and strategies for original Arabic text cannot be applied. The additional intersecting letters and the various cursive and rotational styles in the calligraphy domain require a particular type of segmentation and application of additional methods to handle the different issues of text disordering and to enable potential reading (Azmi et al., 2011). Furthermore, Saberi et al. (2016) defined and explored the ability to read AC decorative arts in the Sul-

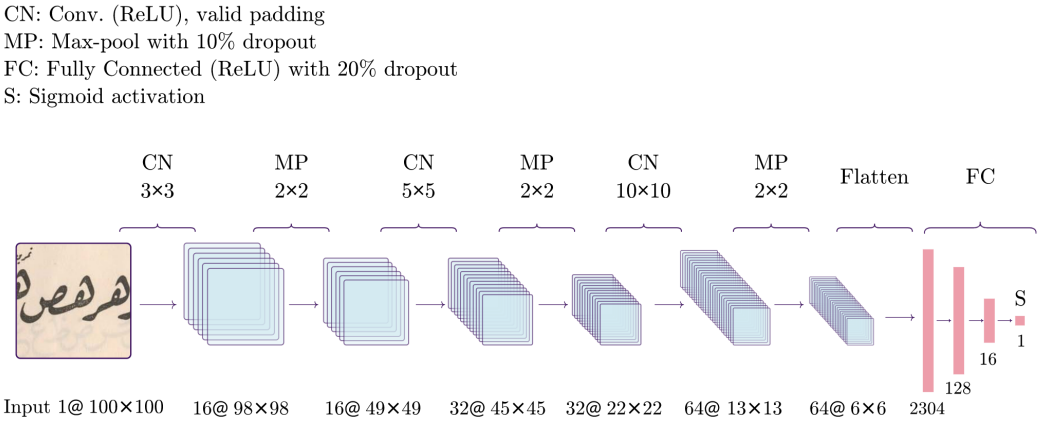


Figure 6: The Final CNN model used during final tests

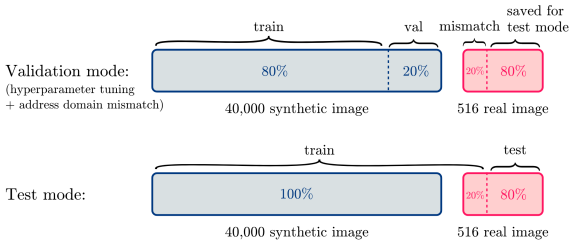


Figure 7: The split strategy

tan Alauddin Masjid, with 409 Arabic reading participants, ranging from 17 to 87 years old. The results show that the participants barely understood the meaning of the inscription, since it was presented as a transformed version of the original Quran quotation with totally different features.

Figure 9 shows an example of the drawing of the Holy Quran's Surah Al-Nas in the Thuluth style by the calligrapher, Hasan Kan'an (bin Talal, 2012). This calligraphy shows the challenges with calligraphic text, with different sizes, and differences in the direction and order of reading of the text all found in the same piece of art. The order of reading moves from right to left but starts from the bottom sentence, which declares the name of the Surah, then moves to the central text inside the circular shape, which is the beginning of the Surah and is then completed by the circular text in the right to left direction. Additional marks and diacritic signs follow and surround the text.

3.1.1 Machine learning approach

Many researchers don't use deep learning for this task because of the lake of data. They use SVM models as an alternative, they also use computer vision image descriptors to extract important features from input images. Let's see some of the mainly used descriptors:

- **SURF features :** Speeded Up Robust Features is mainly inspired on the SIFT descriptor. It uses patches of images and calculates the Hessian matrix for each pixel with gaussian kernel and uses Haar-wavelet response (for orientation resilience). The resulting features are summed by patch and considered as feature vector (4 values per patch) with lower dimension than the SIFT descriptor.

- **HOG features :** Histogram of Oriented Gradients, the simple version of this descriptor is a histogram of the angles of the image's gradients $\left\{\theta = \arctan\left(\frac{G_y}{G_x}\right)\right\}$, for each angle, we associate the number of pixels having the same gradient angle. Other HOG versions use Bins instead of 180 angle or use the sum of the magnitude of the gradients in the histogram instead of the count. It possible to visualize the gradients in the input image as it have been shown in the figure 10.

3.2. Data Generation

As it has been mentioned before, the literature lacks a public benchmark for handwritten Arabic calligraphy style classification. Although researches have used some personal datasets, these datasets suffer from the smallness, the homogeneity and the simplicity, which dispose them from being challenging. Thankfully, Zaid et al. [4] have recently proposed Calliar, which is an online dataset for Arabic calligraphy that contains 2,500 sentences and more than 40,000 strokes. The dataset allows capturing calligraphy in multiple levels ranging from stroke, character, word to sentence level representation. The granularity level allows for

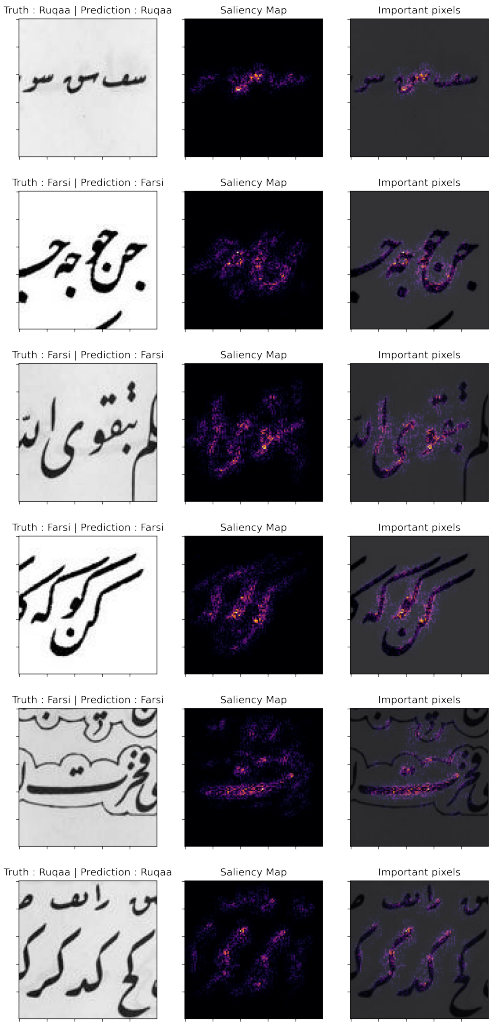


Figure 8: Saliency maps showing important pixels for the class inference.

using the dataset in multiple tasks like classification on the character or the word level. In addition to that, we can use the dataset for calligraphy generation and character recognition. The dataset consists of a wide range of calligraphic styles like Diwani, Thuluth, Farisi, etc. The different styles make the dataset unique because of the complexity of drawing letters in each of the styles.

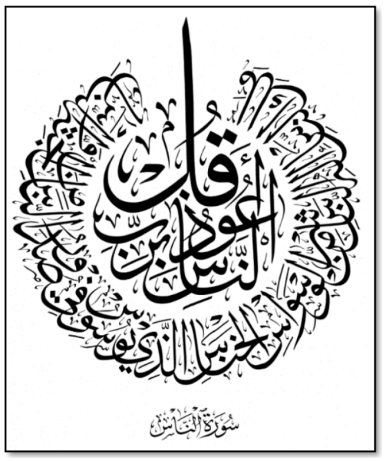


Figure 9: The entire Surah Al-Nas from the Quran; calligraphic art in the Thuluth style, by the calligrapher Hasan Kan'an / Arts College (bin Talal, 2012).

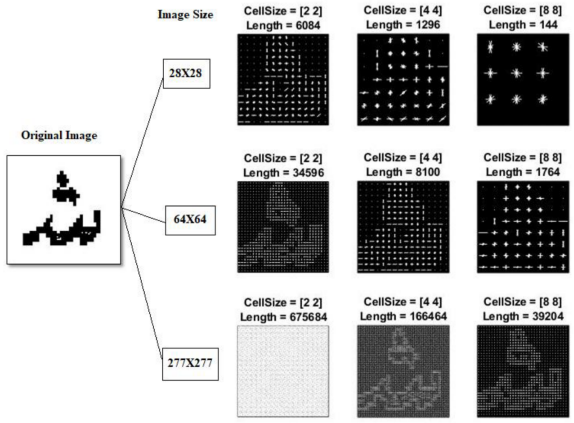
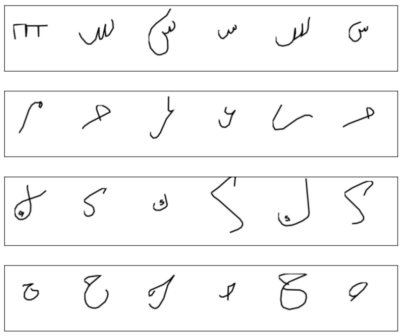


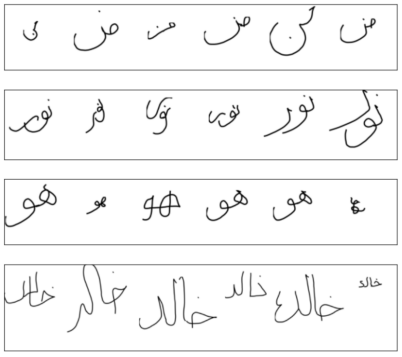
Figure 10: HOG features plotted in the input image

Data visualization and exploration is a necessary technique to assess the quality of the collected dataset. In Arabic script writing, letters are written connected together which causes a variation between a letter in beginning, middle or at the end. Moreover, various calligraphy styles draw characters differently which might cause a single letter to be confused with another. This is much more apparent when we try visualizing individual characters in our datasets. In Figure 11a, we visualize four different letters sampled randomly from our dataset. For example, the letter س, in the first row is written in different variations. The first image of the character is interesting because the character is drawn upside down. This type of drawing mostly happens in Kufic calligraphy. The letter م is also interesting because it can be written with or without a loop depending on the style of calligraphy. The letter ك is unique because it can appear in

either single stroke or double stroke depending in its position on a given word. Similarly in Figure 11b, we see the results of visualizing word level representations. This is important because this proves that our dataset can be used to extract word-level features which can be utilized for word level recognition. The most interesting one is the writing of the word هو which can be written in many variations as seen in the third row. One thing that makes this dataset complex is the freedom of writing in different angles like in the last column of the third row where the word is written vertically.



(a) Sample character variations.



(b) Sample word variations.

Figure 11: Different variations of drawing characters and words.

Since the focus of our initial study is to see how well can recent OCR models detect arabic calligraphy, we start by extracting the characters drawings of each calligraphic sample from the Calliar dataset. We also crop the images to have square shapes of minimum $600px \times 600px$. A preview of the extracted images are shown in the Figure 13

3.3. Approach

The basis for this part of our research is TrOCR (Li et al., 2021), which combines the BERT-style vision transformer BEiT (Bao et al., 2021) with a RoBERTa (Liu et al., 2019) language representation model. BEiT works as an encoder and is pre-trained on the Image-Net-1K (Russakovsky et al., 2015) dataset containing 1.2M images, while RoBERTa serves as a decoder producing the text. Li et al. (2021) used 687M of printed and about 18M of synthetically generated handwritten text lines in English to pre-train the TrOCR model. During this phase, the model learns to extract relevant features from the images and decode them into English text, therefore training the language model from scratch. Li et al. (2021) fine-tuned their pretrained TrOCR instances on "real-world" data, like the IAM dataset (Marti and Bunke, 2002). The IAM dataset consists of handwritten English lines from different authors.

Our research aims to exploit the pre-trained vision and language transformers, hoping that a model fine-tuned on historical manuscripts generalises well enough to be applied to extensive and variable arabic calligraphy collections. We want to test whether we can transfer the "knowledge" about handwriting in the English language TrOCR has acquired from the early modern manuscripts. And since the Arabic language differ a lot from the English language, we chose to warm-start our decoder model with the pretrained AraBERT, while using ViT for the encoder model. See Figure 12

Encoder: We chose for our encoder the ViT model, which applies a pure transformer directly to sequences of image patches to classify the full image. Recently proposed by Dosovitskiy et al. [13], it has achieved state-of-the-art performance on multiple image recognition benchmarks. In addition to image classification, transformer has been utilized to address a variety of other vision problems, including object detection [9], [21], semantic segmentation [19], image processing [10], and video understanding [20]. Thanks to its exceptional performance, more and more researchers are proposing transformer-based models for improving a wide range of visual tasks.

Decoder: The decoder in our vision encoder-decoder is AraBERT. Which was released by Antoun et al. (2020) and was among other models compared to mBert. AraBERT achieved state-of-the-art performance on most tested Arabic NLP tasks. The models were trained on news articles manually scraped from Arabic news websites and several publicly available large Arabic corpora. One of the corpora is named OSCAR (Open Super-large Crawled Aggregated Corpus), not to be confused with the image captioning model OSCAR (Object-Semantics Aligned Pre-training). In total, the dataset consists of $\sim$ 77 GB of text. There are several versions of AraBERT available.

3.4. Experimental Setting

The pretrained weights used for our TrOCR variant come from `aubmindlab/bert-base-arabertv2` and `google/vit-base-patch16-224-in21k` from the HuggingFace hub for our encoder and decoder respectively. We trained the model for 500 epochs with

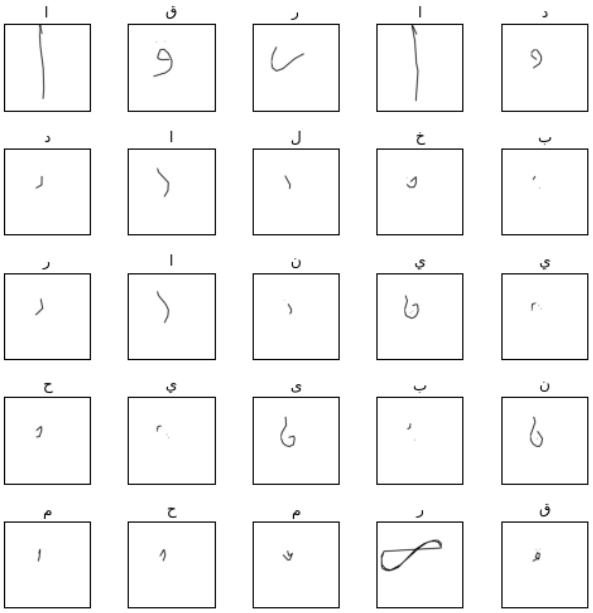
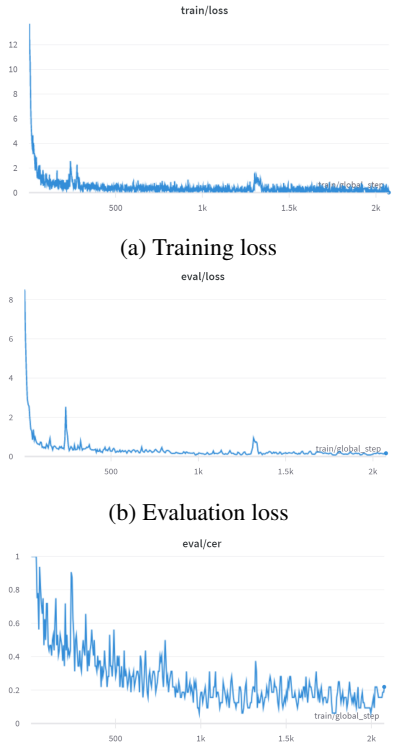


Figure 13: Preview of the processed dataset.



(a) Training loss

(b) Evaluation loss

(c) Evaluation character error rate (CER)

Figure 14: Training curves

a batch size of 8 on a Geforce 3090. We evaluated the model every 5 steps on an evaluation dataset constituted of 64 samples. And to avoid overfitting on our limited data, we set an early stopping with a patience of 250 steps. The training curves are shown in the figures 14

We obtained a surprising character error rate (CER) of 21%, which was unexpected for a pretrained encoder model that never saw cursive text before, let alone arabic text. To further investigate what the ViT encoder has learnt we plot the average attention weights in the last Transformer block in the appendix A.1. We can see that some heads tend to focus mostly on some specific strokes shapes for some letters, while some heads focus on wider scopes in the images.

Since, the calligraphic writing for some letters can hardly be distinguished, we also plot the confusion matrix for the

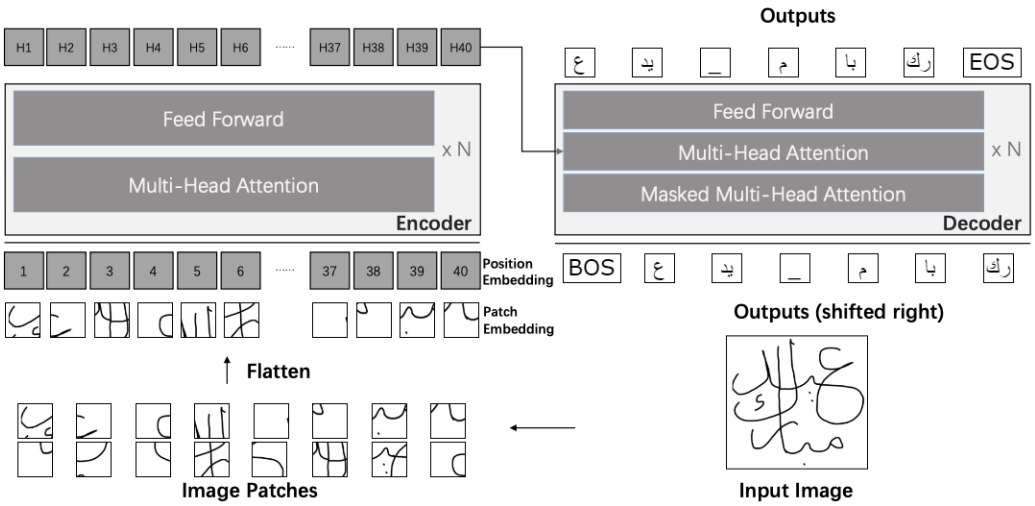


Figure 12: Proposed TrOCR model.

same experiment, see Figure 15. This confirms that the model has a hard time to distinguish some similar letters like ل , ر and ا.

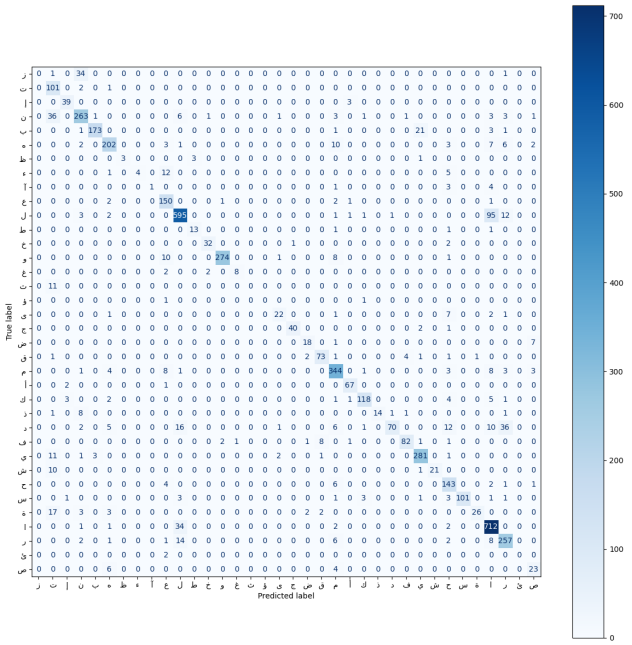


Figure 15: Confusion Matrix.

3.5. Results and discussion

3.5.1 Minimal pre-processing of data

During this project, we applied no pre-or post-processing of the Arabic raw text. This could have a negative effect on the performance of our models and the final results. In their Arabic image captioning work, ElJundi et al. (2020) writes: "It is crucial to clean and pre-process our data before feeding it to any model because 'garbage in, garbage out'..."

They followed Arabic pre-processing techniques recommended by Shoukry and Rafea (2012):

1. Remove (harakat) diacritics.
2. Normalize the hamza (ء) on characters (for example to distinguish between a glottal stop and a mere vowel, hamza is usually added to letter Alif (ا) diacritically, either above (أ) or below (إ)).
3. Normalize some word ending characters, such as taa marbouta (ة) and ya' maqsoura (ي).
4. Remove punctuation as well as non-Arabic letters.

   It is hard to say how this text processing scheme applied on our work would affect the final scores, but we think that a pre-processing scheme similar to the one above could give our models better performance. From the context of pre-processing point 2 ., our candidate caption output already seems to be hamza-normalized (i.e. all {أ or إ → ا), while reference captions still contains extra hamzas on them. During evaluation, this of course affects the mean CER scores negatively, since the similarity function between symbols أ, إ and ا produces CER scores more than 0.

Another kind of Arabic text processing is sub-word units segmentation used in training some of the AraBERT models released by Antoun et al. (2020). The authors reduce the model vocabulary by segmenting words into into stems, prefixes and suffixes. For instance, "اللغة Alloga" becomes "ة لغ ال". Since we chose `bert-base-arabertv2`, which is trained on non-segmented text, we did not use sub-word segmentation. Nonetheless, it would be interesting to see how segmentation applied to the candidate and reference captions would affect the evaluation scores.

4. Future Work

In addition to the above-mentioned examples, the dataset could be used for sketch generation as a creative application. Most of the research in the literature that deals with sketch drawing and text-to-image, consider English like GANwriting [17], Scrabble-GANs [14], DF-GANs [18], BézierSketch [11], sketchRNN [15] and DoodlerGAN [16]. This field of research is very important because it combines multiple modalities ranging from natural language processing (NLP) , generative adverserial networks (GANs) and creative applications. In the literature, there are hundreds of papers published in each year for each of those fields but due to the lack of proper datasets, there are no real advancements in making systems that deal with all of them especially for Arabic. We believe that this dataset can fill this gap.

4.1. Bag of letters

The character recognition model could be used not as a text generator, but as character counter. This will be done by replacing the decoder by a 2 fully connected layer with an output of 36 (the possible characters in our vocabulary), the problem will be formulated a multi dimensional regression to count the number of the characters in each image. This bag of letters could be used as TF-IDF vector, a search will be performed on the main possible verses and texts usually used in calligraphy to detect the nearest neighbor to the bag of letters vector. It is also possible to augment the size of the output layer to predict N-grams, this would be easier for the model as there are usually some easy to spot words in the calligraphy.

4.2. End to End word and sentence recognition

It is possible to use an end to end word [2] or sentence [3] recognition. We have already trained the decoder encoder model for theses tasks and got the CER error of 0.4 and 0.6 respectively. These results might need additional processing to reduce the error.

The next step would be to use prior knowledge of the possible words and sentences (Mainly from religious resources) to correct the model output. We have seen that sometimes the model yields good sentences but with one or two incorrect words, so it is possible to search for the nearest known sentence for correction using CER as a distance. It is also possible to retrain the decoder's tokenizer to be specific to words and sentences used in arabic callligraphy.

5. Conclusion

As we previously seen, it is possible to detect the Arabic calligraphy style using simple CNN networks of about half a million parameter with high score. However, there are many limitations in this work, first of them is the lake of data and low number of the considered classes. The second application of character recognition promises good results if it is used with the bag of letters technique or N-grams to infer the written calligraphy.

References

[1] ithaca. https://ithaca.deepmind.com/.

[2] Mahmoud Aslan arabic font classification. https://mhmoodlan.github.io/blog/arabic-font-classification#4-data. Accessed: 2022-04-10.

[3] Zaid Alyafeai and Maged Al-Shaibani. ARBML: Democritizing Arabic natural language processing tools. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, pages 8–13, Online, Nov. 2020. Association for Computational Linguistics.

[4] Zaid Alyafeai, Maged S Al-shaibani, Mustafa Ghaleb, and Yousif Ahmed Al-Wajih. Calliar: An online handwritten dataset for arabic calligraphy. *arXiv preprint arXiv:2106.10745*, 2021.

[5] E. Ataer and P. Duygulu. Retrieval of ottoman documents," in proceedings of the 8th acm international workshop on multimedia information retrieval,, 2006.

[6] Mohd Sanusi Azmi, Khairuddin Omar, Mohammad Faidzul Nasrudin, Azah Kamilah Muda, Azizi Abdullah, and Khadijah Wan Mohd Ghazali. Features extraction of arabic calligraphy using extended triangle model for digital jawi paleography analysis. *Int. J. Comput. Inf. Syst. Ind. Manag. Appl*, 5:696–703, 2013.

[2] https://wandb.ai/nouamanetazi/TrOCR/runs/17jdaji1?workspace=user-nouamanetazi

[3] https://wandb.ai/mustapha/TrOCR/runs/10utsitc?workspace=user-mustapha

[7] Beckman I. Kedem K. et al Bar-Yosef, I. character extraction, and writer identification of historical hebrew calligraphy documents. In *character extraction, and writer identification of historical Hebrew calligraphy documents*, volume 1, pages 89–99, 2007.

[8] Bilal Bataineh, Siti Norul Huda Sheikh Abdullah, and Khairudin Omar. Arabic calligraphy recognition based on binarization methods and degraded images. In *2011 International Conference on Pattern Analysis and Intelligence Robotics*, volume 1, pages 65–70, 2011.

[9] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[10] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12299–12310, 2021.

[11] Ayan Das, Yongxin Yang, Timothy Hospedales, Tao Xiang, and Yi-Zhe Song. Béziersketch: A generative model for scalable vector sketches. In *European Conference on Computer Vision*, pages 632–647. Springer, 2020.

[12] Nachum Dershowitz and Andrey Rosenberg. Arabic character recognition.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[14] Sharon Fogel, Hadar Averbuch-Elor, Sarel Cohen, Shai Mazor, and Roee Litman. Scrabblegan: Semi-supervised varying length handwritten text generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4324–4333, 2020.

[15] Songwei Ge, Vedanuj Goswami, C Lawrence Zitnick, and Devi Parikh. Creative sketch generation. *arXiv preprint arXiv:2011.10039*, 2020.

[16] Rachid Benslimane Ilahm Chaker and Mostafa Harti. Creation of al mabsout moroccan fonts format. *n 2011 Colloquium in Information Science and Technology,*, 2011.

[17] Lei Kang, Pau Riba, Yaxing Wang, Marçal Rusiñol, Alicia Fornés, and Mauricio Villegas. Ganwriting: content-conditioned generation of styled handwritten word images. In *European Conference on Computer Vision*, pages 273–289. Springer, 2020.

[18] Ming Tao, Hao Tang, Songsong Wu, Nicu Sebe, Xiao-Yuan Jing, Fei Wu, and Bingkun Bao. Df-gan: Deep fusion generative adversarial networks for text-to-image synthesis. *arXiv preprint arXiv:2008.05865*, 2020.

[19] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF conference on*

computer vision and pattern recognition, pages 6881–6890, 2021.

[20] Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. End-to-end dense video captioning with masked transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8739–8748, 2018.

[21] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

A. Appendix

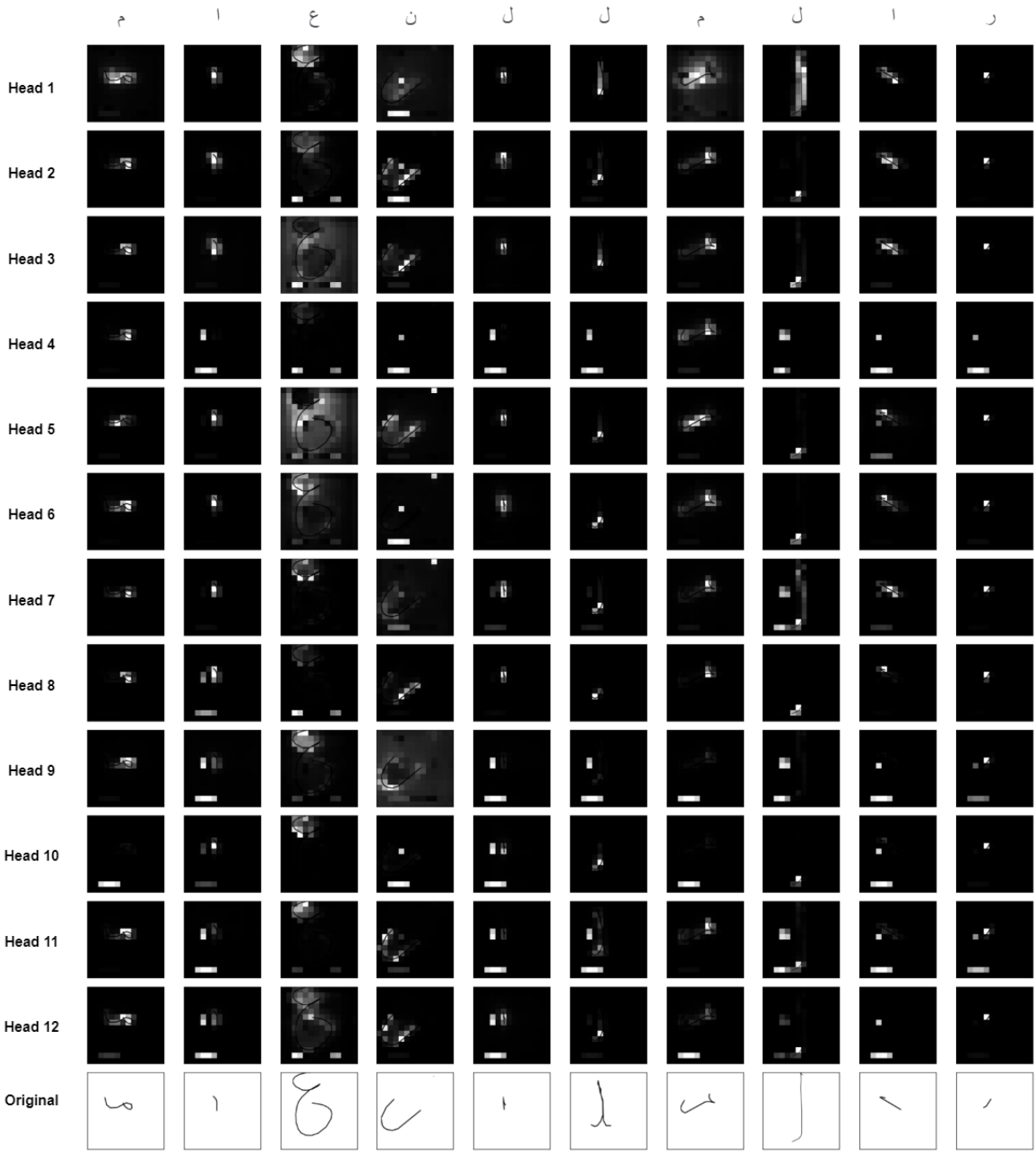


Figure A.1: Average attention weights in the 12th Transformer block for multiple inputs.